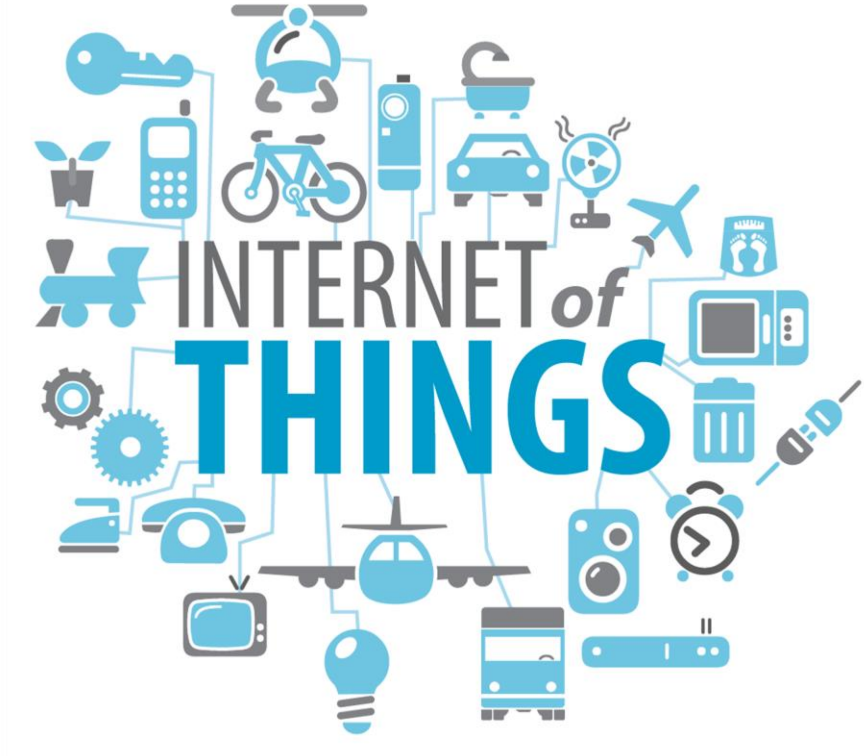
Version 2.0

TECHNICAL NOTE

Beandevice® Wilow - Using MQTT with Node Red



www.beanair.com

Document			
Document number		**Version**	V2.0
External Reference	TN-RF-19	**Publication date**	31/03/2020
Author	Aymen Jegham		
Internal Reference		**Project Code**	N.A.
Document Name	***Beandevice® Wilow - Using MQTT on Node Red***		

Validation			
Function	**Recipients**	**For Validation**	**For information**
Reader	Yosri Jaouadi		X
Author	Aymen Jegham	X	

Mailing list			
Function	**Recipients**	**For action**	**For Info**
Staffer 1	Yosri Jaouadi	X	
Staffer 2			X

Updates

Version	Date	Author	Evolution & Status
1.0	02/01/2018	Aymen Jegham	First version of document
1.0.1	10/05/2019	Mohamed Bechir Besbes	Weblinks Update
2.0	31/03/2020	Habib Jomaa	• Change the name of the document • configuration section (images and descriptions) using Beanscape version 3.2.0.15. • Add another MQTT test with beanscape. • Update the table of acquisition modes. • Add dynamic example section using a Streaming mode as an example.

Contents

List of Figures

Disclaimer

- The information contained in this document is the proprietary information of Beanair.
- The contents are confidential and any disclosure to persons other than the officers, employees, agents or subcontractors of the owner or licensee of this document, without the prior written consent of Beanair Ltd, is strictly prohibited.
- Beanair makes every effort to ensure the quality of the information it makes available. Notwithstanding the foregoing, Beanair does not make any warranty as to the information contained herein, and does not accept any liability for any injury, loss or damage of any kind incurred by use of or reliance upon the information.
- Beanair disclaims any and all responsibility for the application of the devices characterized in this document, and notes that the application of the device must comply with the safety standards of the applicable country, and where applicable, with the relevant wiring rules.
- Beanair reserves the right to make modifications, additions and deletions to this document due to typographical errors, inaccurate information, or improvements to programs and/or equipment at any time and without notice.
- Such changes will, nevertheless be incorporated into new editions of this document.

1. TECHNICAL SUPPORT

For general contact, technical support, to report documentation errors and to order manuals, contact ***BEANAIR® Technical Support Center*** (BTSC) at:

tech-support@Beanair.com

For detailed information about where you can buy the Beanair equipment/software or for recommendations on accessories and components visit:

www.Beanair.com

To register for product news and announcements or for product questions contact BEANAIR®'s Technical Support Center (BTSC).

Our aim is to make this user manual as helpful as possible. Please keep us informed of your comments and suggestions for improvements. Beanair appreciates feedback from the users.

2. VISUAL SYMBOLS DEFINITION

Visual	*Definition*
	***Caution or Warning** – Alerts the user with important information about Beanair wireless sensor networks (WSN), if this information is not followed, the equipment /software may fail or malfunction.*
	***Danger** – This information MUST be followed if not you may damage the equipment permanently or bodily injury may occur.*
	***Tip or Information** – Provides advice and suggestions that may be useful when installing Beanair Wireless Sensor Networks.*

3. AIM OF THE DOCUMENT

The aim of this document is to demonstrate a simple integration of the BeanDevice® Wilow in the Internet of things ecosystem using cutting-edge technology, this will be very important for a countless possibilities of measurements, collecting, analyzing and processing the data .

This document is not intended to deliver an extreme precise step-by-step tutorial but will focus more on the specification of the BeanDevice® MQTT frame and the basic use of the IBM Node-RED Internet of things flow based development platform.

1. OVERVIEW

The idea is to introduce the BeanDevice to the internet of things using the MQTT protocol and the IBM flow based Node-RED, data can be collected locally or on the cloud using the IBM bluemix platform.

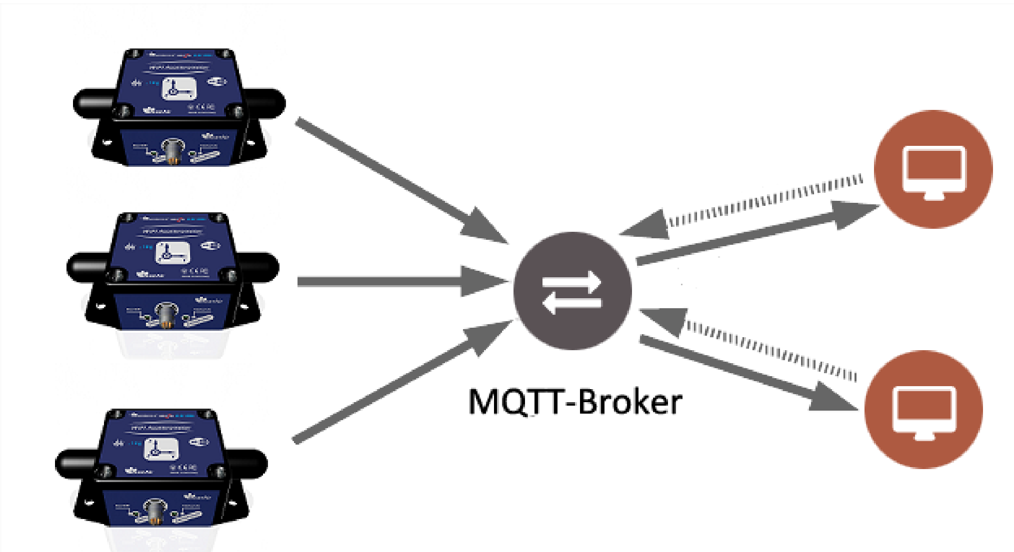


Figure 1: System diagram

We can install and use local MQTT broker or use a free of cost online broker (limited). The BeanDevice will publish data to all subscribed devices on its topic, and we can publish configuration (change acquisition mode, restart BeanDevice ...set sleep mode) to a subscribed BeanDevice.

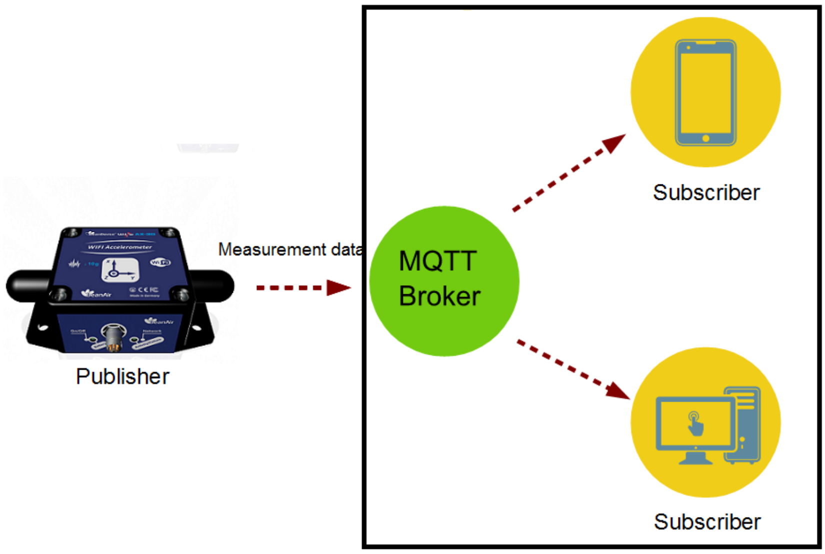


Figure 2: Data collection

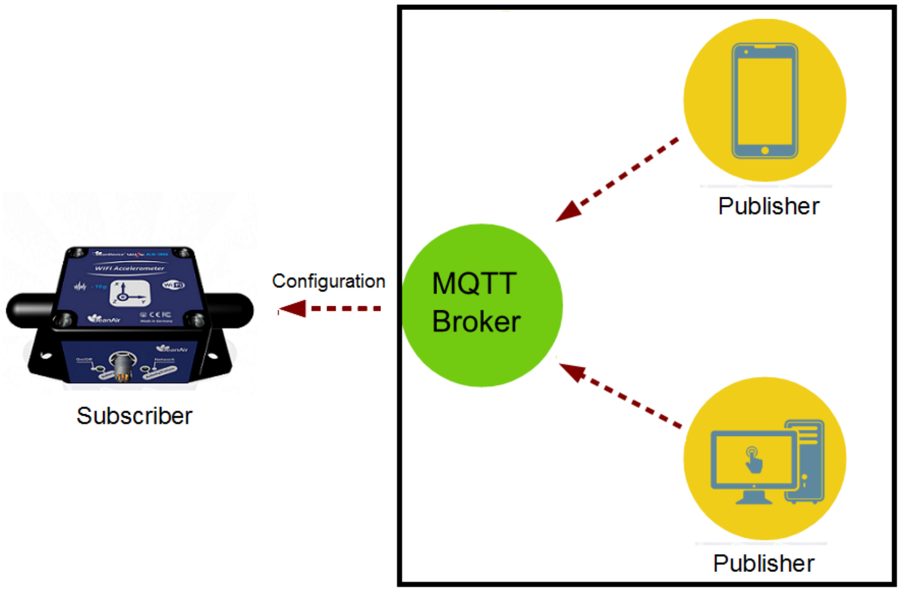


Figure 3: BeanDevice Configuration over MQTT

2. INSTALATION AND ENVIRONMENT CONFIGURATION

2.1 NETWORK

To get started using Our BeanDevice Wilow over MQTT and before starting the configuration we need to install an MQTT broker on any embedded computer or SBC of your choice(Raspberry PI,Beaglebone black,..)Alternatively, even use a windows system (like in this example), also we can simply use an online broker (free with limits), next we build our WiFi network and make sure we have the network architecture as shown in the figure below.

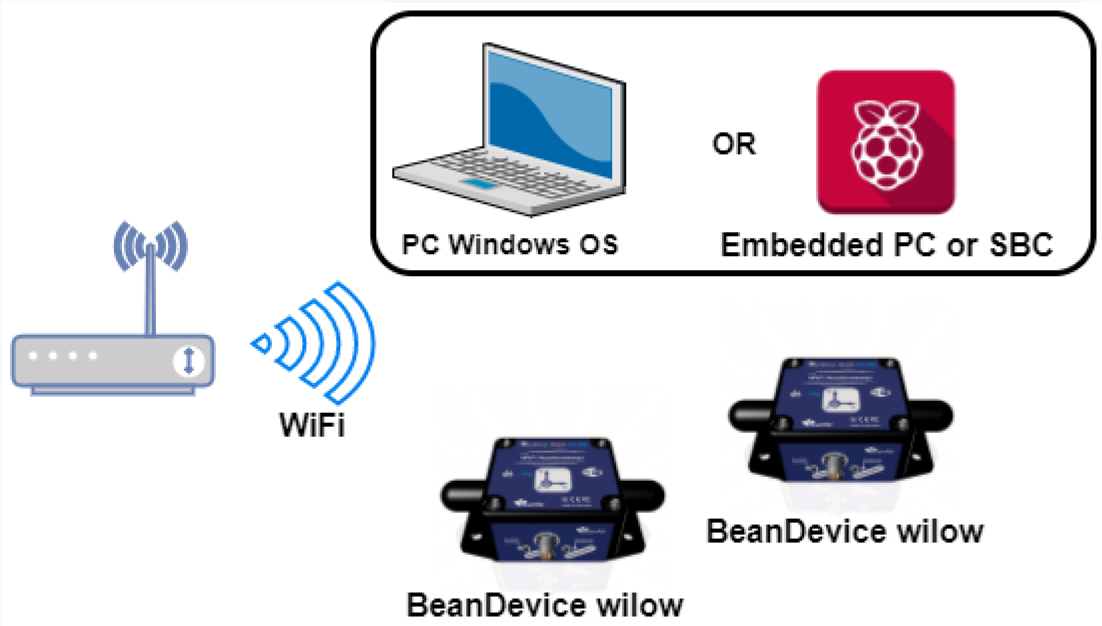
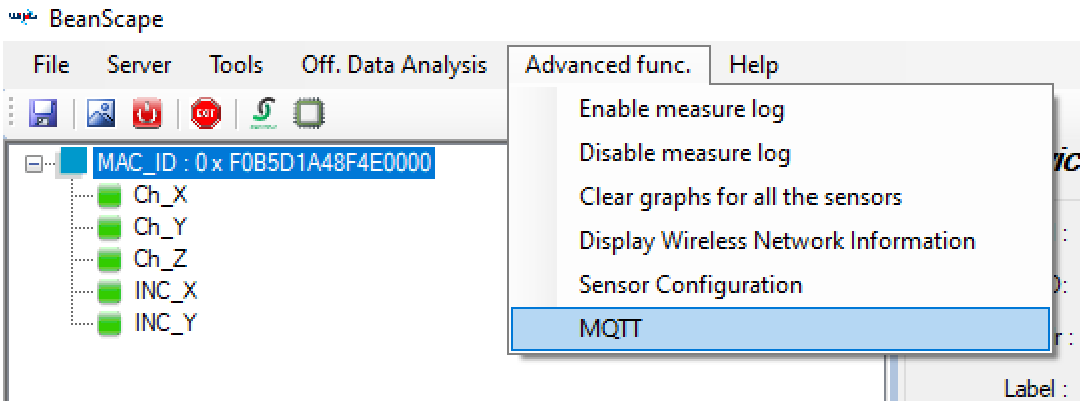


Figure 4: Local network configuration

2.2 CONFIGURATION

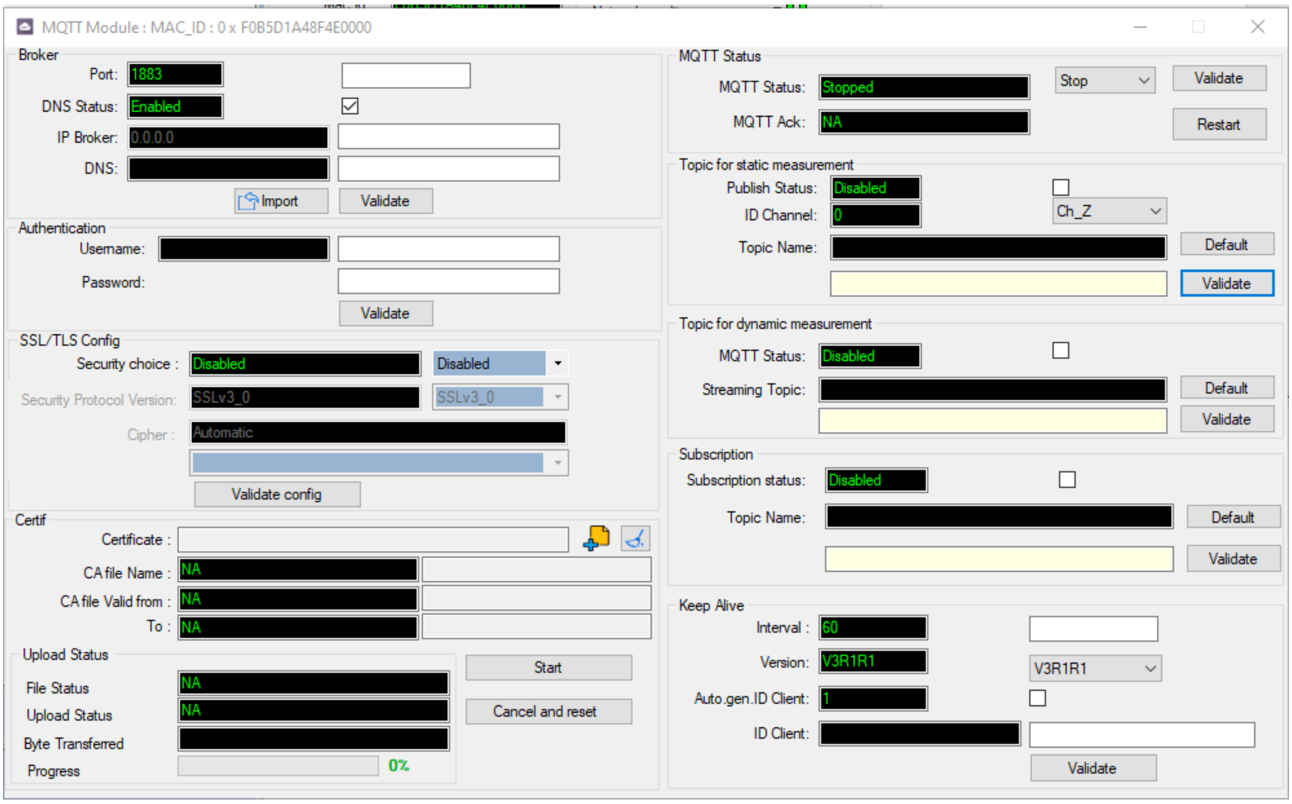
In order to start the MQTT communication we have to setup the MQTT configuration using BeanScape, after connecting the BeanDevice to the network (find more details in our YouTube video: getting started with BeanDevice Wilow).

Select your BeanDevice and scroll down to MQTT in the BeanDevice tab.

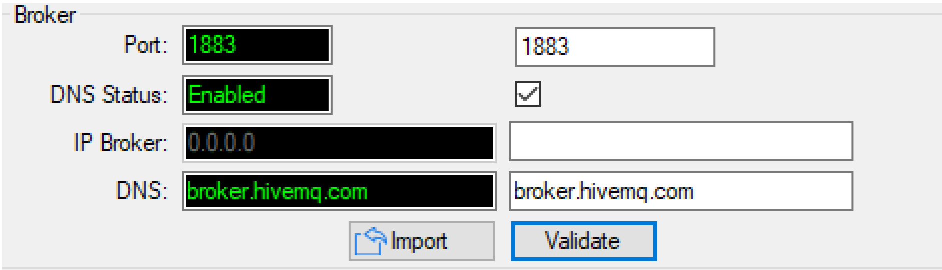


A new window will pop up and it is where we will configure the BeanDevice MQTT module.

To make things simple we will not use the security feature (SSL/TLS and Certif options).



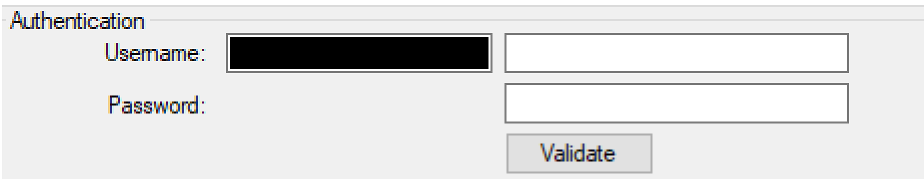
Broker



- ***Port***: TCP/IP port to use with MQTT .1883 and 8883(secured port, over SSL/TLS) are the reserved (default) ports for MQTT.
- ***DNS Status***: check it if you want to use your broker DNS otherwise uncheck it if you want to use your broker ip address.
- ***IP Broker***: enter your broker Ip address (make sure to uncheck the DNS Status).
- ***DNS***: enter the DNS(domain name server) of your Broker (make sure to check the DNS Status)
- ***Import button***: Import saved configuration (last used configuration).
- ***Validate***: confirm and save your broker configuration.

Authentication

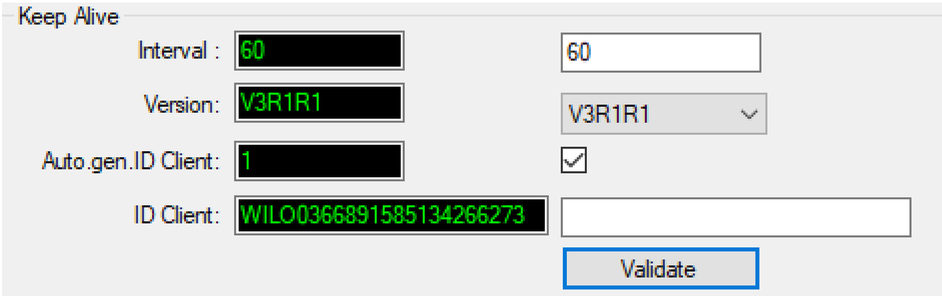
MQTT broker can be configured to require client authentication using a valid username and password before a connection is permitted.



- ***Username:*** specify your user name
- ***Password:*** enter your password
- ***Validate***: save your configuration.

Keep alive

The keep alive functionality assures that the connection is still open and both broker and client are connected to one another



- ***Interval***: The interval is the longest possible period of time, which broker and client can endure without sending a message.
- ***Version***: MQTT Protocol version
- ***Auto.gen.ID Client***: check for auto generate a Client ID
- ***Client ID***: Enter your client ID manually (make sure to uncheck Auto_gen.ID Client)
- ***Validate***: save your configuration.

Topic for static measurement

The topic is a string used by the broker to filter messages for each connected client.

"Topics for static measurement" section is only for LowDutyCycle and Alarm modes.

In static mode (LDC or Alarm) each sensor in the beandevice will publish its measurements to a specific and well reserved topic.

In our case we will subscribe to those Topics to receive the static measurements from each sensors.

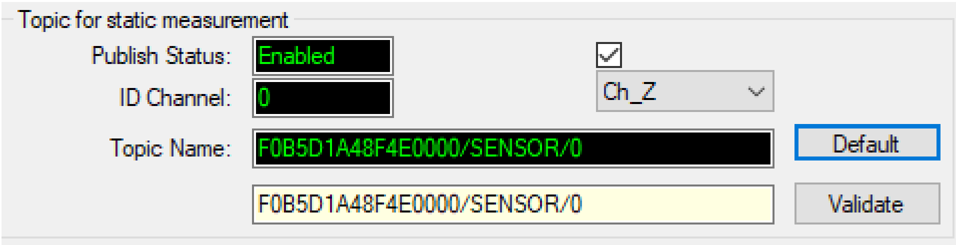
For better and easy use, Topic names are not configurable and they are as follow:

[BeanDevice_MAC-ID]/SENSOR/[sensor-ID]

For Example: F0B5D1A48F4E0000/SENSOR/0

F0B5D1A48F4E0000: Beandevice mac id

0: channel Z



- ***Publish Status:*** check it to enable publishing.
- ***ID Channel:*** channel identification, select sensor from the list.
- ***Topic Name:*** display the used Topic name to publish measurement to (not configurable).
- ***Default:*** to set a default configuration. You need to click this button to set the Topic name.
- ***Validate:*** save your configuration.

Topic for dynamic measurement

Here you enable the Topic for dynamic measurements and it works only for the streaming, S.E.T and Shock Detection modes.

The beandevice will publish all measurement for all sensors to a single Topic.

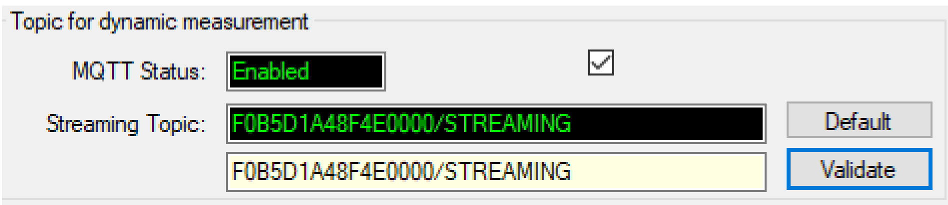
Again, the topic name is not configurable and you can only enable or disable this option.

The topic format is as follow:

[BeanDevice_MAC-ID]/STREAMING

For Example: F0B5D1A48F4E0000/STREAMING

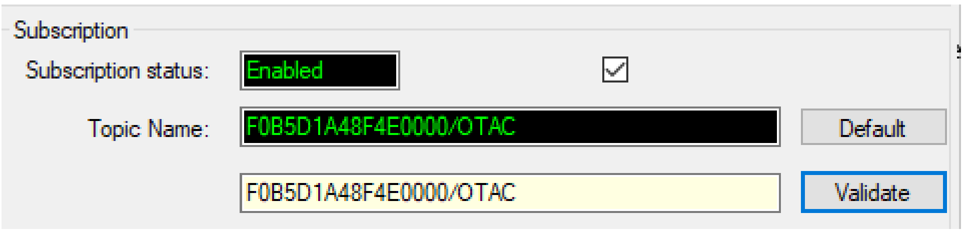
F0B5D1A48F4E0000: beandevice ID



- ***MQTT_status:*** check it to enable publishing
- ***Streaming Topic:*** display the used Topic name to publish measurement to (not configurable).
- ***Default:*** to set the default configuration. You need to click this button to set the Topic name.
- ***Validate:*** save your configuration.

Subscribe

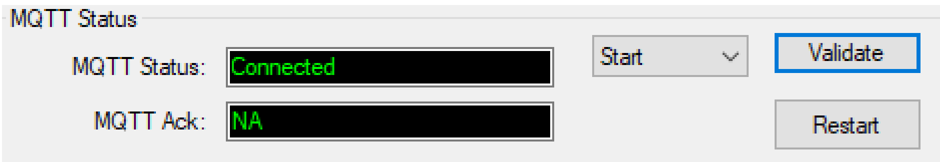
The BeanDevice will subscribe to a another MQTT client who will publish configuration messages,



- ***Subscription status:*** check it to enable subscribing.
- ***Topic Name:*** Field to enter your topic's name to subscribe to.
- ***Default:*** to set the default configuration. You need to click this button to set the Topic name.
- ***Validate:*** save your configuration.

MQTT STATUS

Here you can check your MQTT different status, connected, stopped, connecting or disconnecting and can start/restart your connection from here.

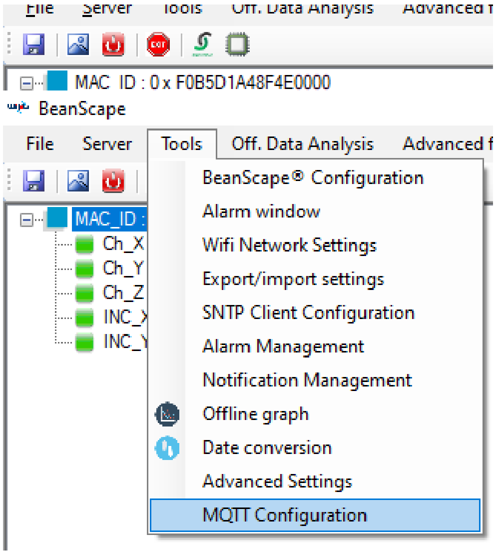


- ***MQTT Status:*** shows the current status of the MQTT module:
  - Connecting: trying to establish a connection
  - Connected: connection established
  - Disconnecting: disconnecting the Client
  - Stopped: the connection is stopped
- ***Start/Stop:*** select to start or to stop your MQTT Client connection
- ***Restart:*** restart your connection

2.3 TESTS

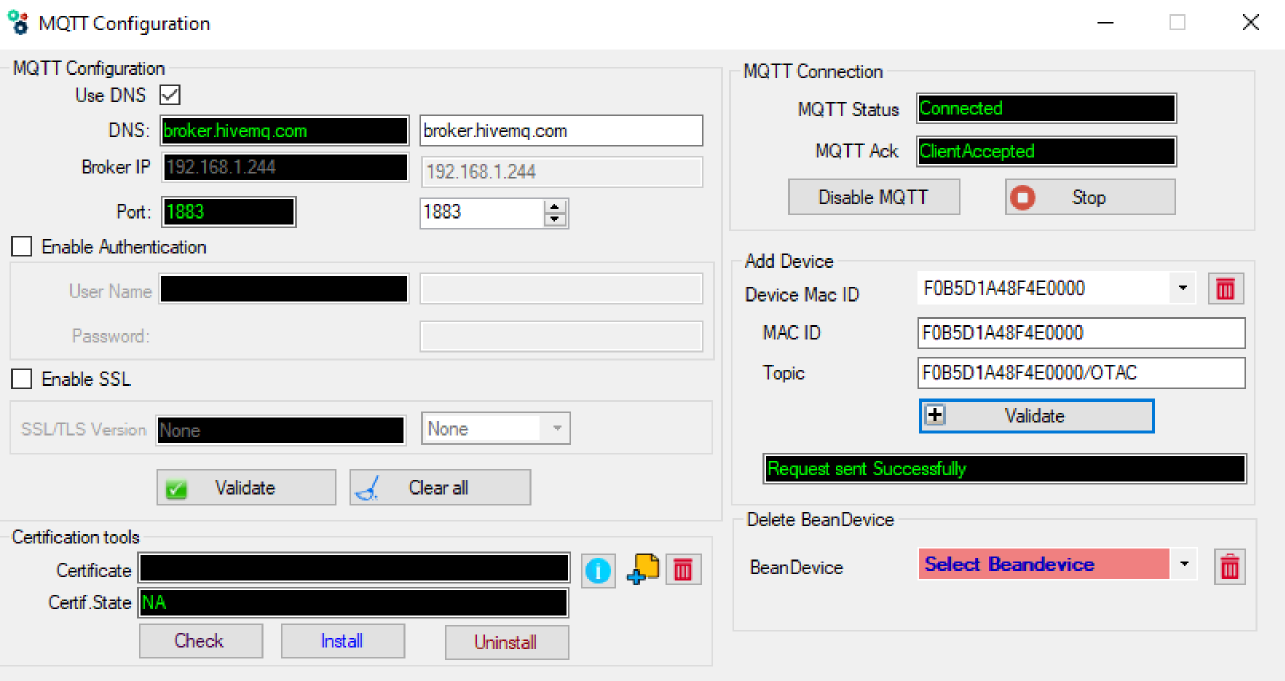
After configuring the BeanDevice MQTT module you can switch Beanscape to MQTT mode and connect to your broker.

Select your beandevice -> go Tools -> click MQTT Configuration:

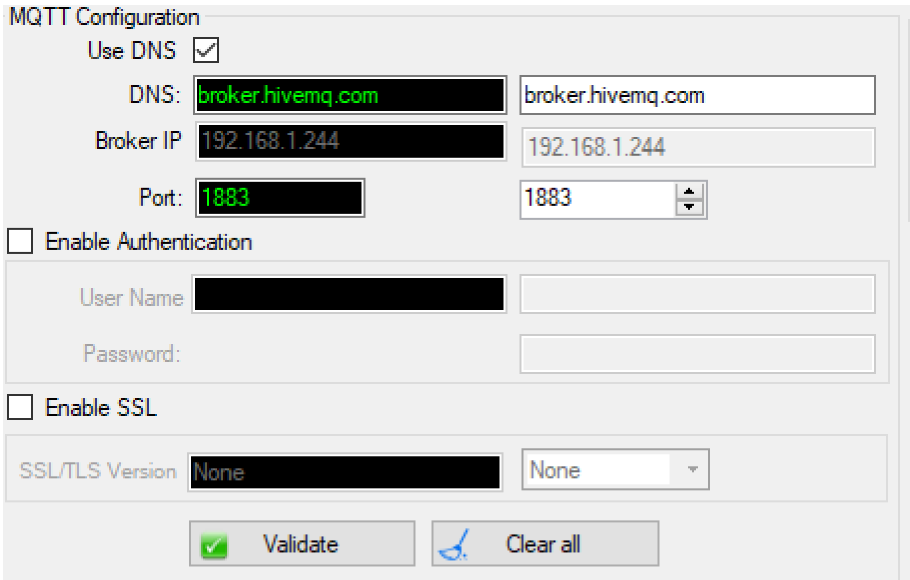


A new window will pop up to configure the Beanscape MQTT.

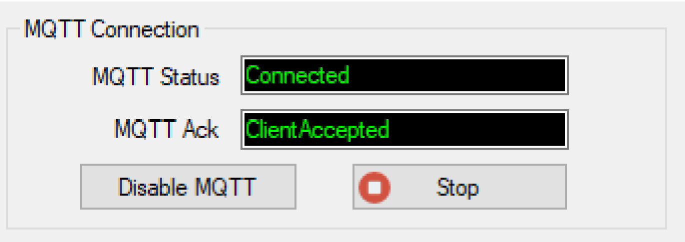
Because the broker isn't configured to use any security feature, don't enable the Authentication, SSL and certification tools options.



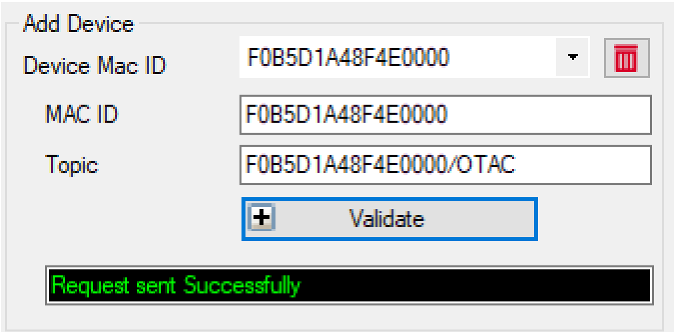
- ✓ Set your broker configuration in this MQTT configuration section (make sure to use the same broker (ip or DNS) and port you used to configure the beandevice), after that click validate:



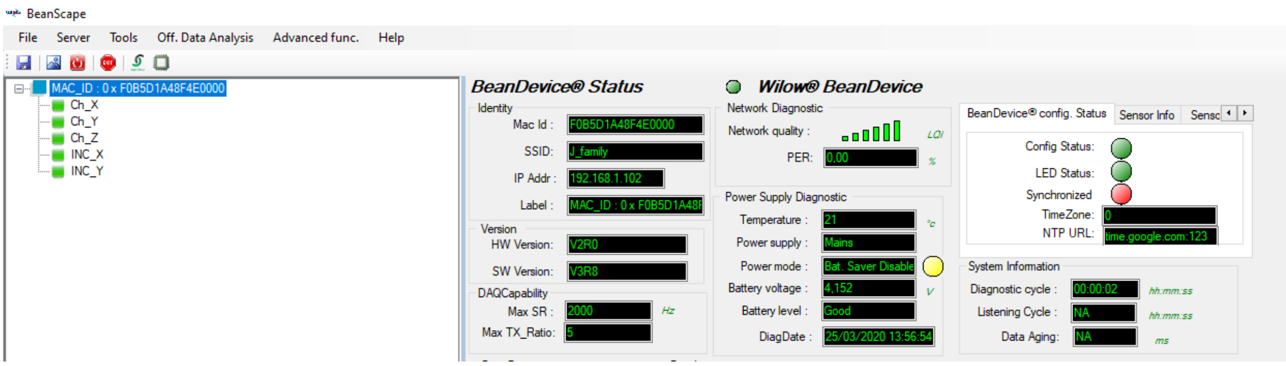
- ✓ Enable MQTT mode by using the Enable (Disable) MQTT button in the MQTT connection section, and then click start:



✓ Add your beandevice to the list of devices by typing the device mac-ID in the "MAC ID" field and set the topic name in the "Topic" field to the same topic you used for the subscription section in the beandevice MQTT configuration.



✓ If your setup is correct and followed the exact same steps, you will be able to see your beandevice added to the list of device, and you can configure it and receive measurement.



✓ Set your BeanDevice to LowDutyCycle acquisition mode and check the graph (make sure to enable the channel's topic for the static measurement):

Another way to test your setup and your BeanDevice MQTT configuration is to use a desktop MQTT test client ([Download for free from here]) to subscribe to one of its static measurement topics after setting your BeanDevice to LowDutyCycle acquisition mode.

Configure the desktop MQTT Client MQTT.FX to connect to **broker.hivemq.com** broker which is the same broker the beandevice connected to (use a Client ID of your choice) and subscribe to the F0B5D1A48F4E0000/SENSOR/0 Topic so we can read the payload sent from the BeanDevice sensor 0 (which is channel Z)

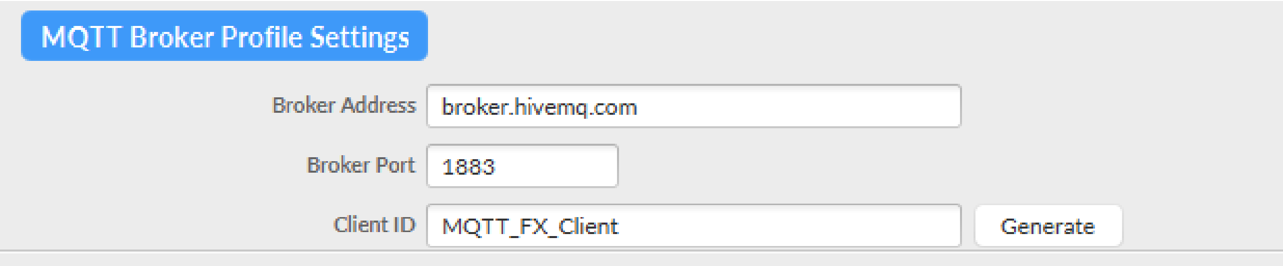


Figure 5: MQTT.fx Broker profile

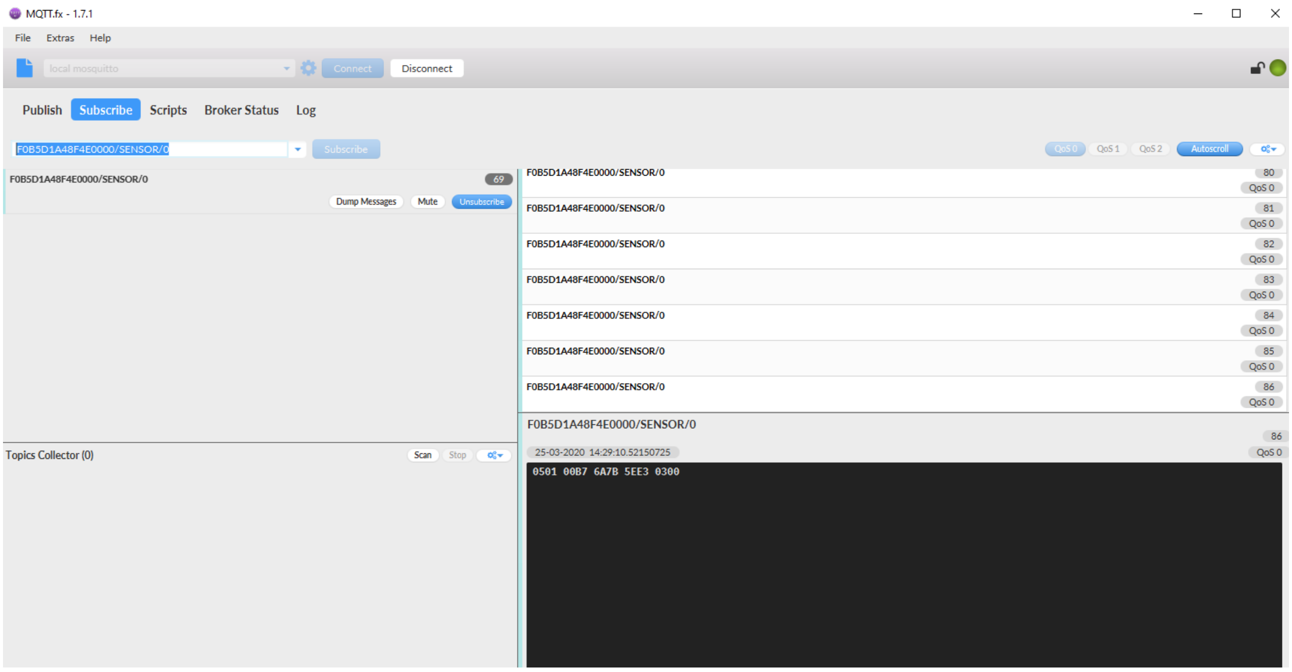


Figure 6: Raw payload displayed on Mqtt.fx

2.4 NODE-RED

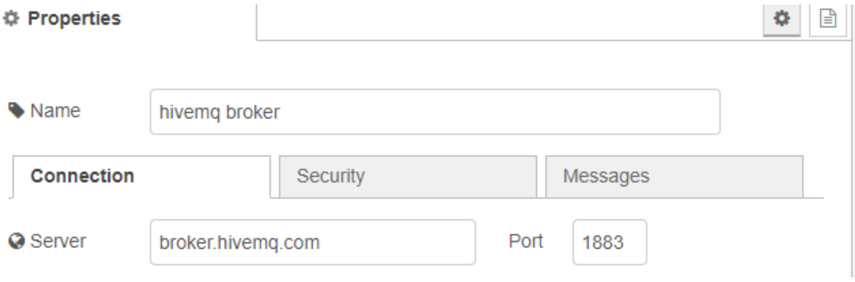
2.4.1 Getting started

For anyone who wants to start using Node-Red to collect data from a BeanAir Wilow WSN we provides you examples for both, static and dynamic modes with visual results.

2.4.2 Static mode

In order to make the data readable and manageable we will proceed with an example based on node-RED, a flow-based development tool by IBM (more info and installation guide here) to collect data from a device in Low Duty Cycle mode which is one of the static modes.

In node-RED, we configure an MQTT node to listen to our TOPIC



Set broker configuration

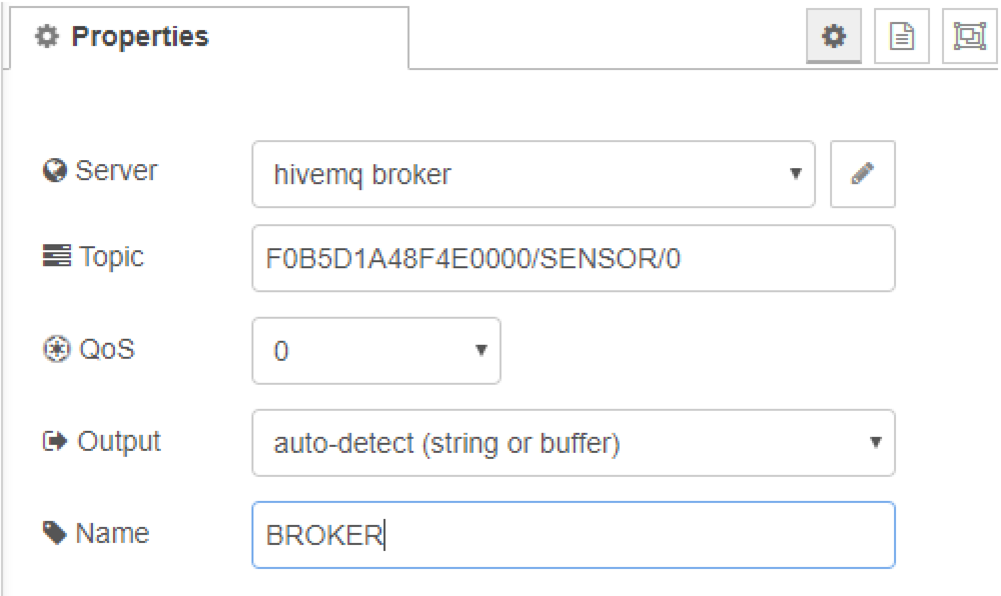


Figure 7 :MQTT node configuration

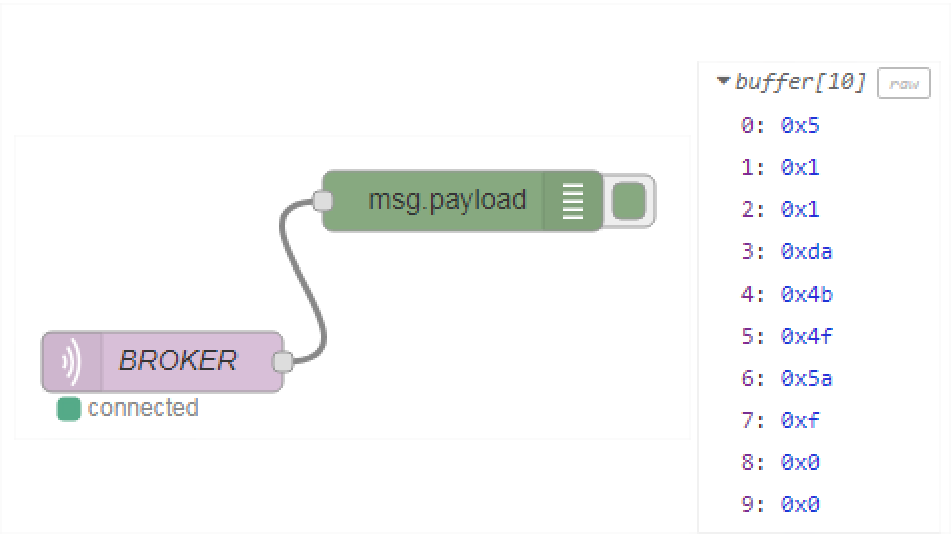


Figure 8 :A simple debugger node displays the raw payload frame

2.4.2.1 Payload

The MQTT frame sent from the BeanDevice® Wilow® will be composed of ten Bytes in the LowDutyCycle mode for example, and the content will be distributed as in the table below:

<table>
<tr><th colspan="3">Data meaning</th><th>Size</th></tr>
<tr><td colspan="3">Device Type</td><td>1 byte</td></tr>
<tr><td colspan="3">Acquisition type (Default 0x01)</td><td>1 byte</td></tr>
<tr><td colspan="3">Channel Id</td><td>1 byte</td></tr>
<tr><td colspan="3">Date in Unix time format (LSB First)</td><td>4 bytes</td></tr>
<tr><td rowspan="4">Data sample measured (LSB First)</td><td colspan="2">Byte[0] data bits</td><td>1 byte</td></tr>
<tr><td colspan="2">Byte[1] data bits</td><td>1 byte</td></tr>
<tr><td rowspan="2">Byte[2]</td><td>Sign bit</td><td>8th bit</td></tr>
<tr><td>data bits</td><td>7 bits</td></tr>
</table>

Table 1: LowDutyCycle MQTT frame content

Example of a LowDutyCycle MQTT payload:

```
0: 0x5
1: 0x1
2: 0x1
3: 0xda
4: 0x4b
5: 0x4f
6: 0x5a
7: 0xf
8: 0x0
9: 0x0
```

- Device type will be identified in the Byte (0) according to the figure below:

Device type	Value
AX 3D	0x01
HI INC MONO	0x02
HI INC BI	0x03
X- INC MONO	0x04
X-INC BI	0x05
AX 3DS	0x06

Figure 9 : BeanDevice type

```
0: 0x5----> X-INC
```

- The Byte (1) will tell us about the Acquisition mode:

Data Acquisition type	*Value*	*Description*
LDCDA mode	*0x01*	*The Id of the Low Duty Cycle Data Acquisition mode*
Alarm mode	*0x02*	*The Id of the Alarm Data Acquisition mode*
Streaming mode	*0x03*	*The Id of the Streaming Data Acquisition mode*
Shock Detection mode	*0x04*	*The Id of the Shock Detection mode*
LDC Math Result	*0x05*	*The Id of the Low Duty Cycle Math Result*
SET mode	*0x06*	*The Id of the SET (STREAMING WITH EVENT TRIGGER) mode*
Dynamic Math Result	*0x07*	*The Id of the Dynamic Math Result*

Figure 10 : Acquisition modes

```
1: 0x1---->LowDutyCycle Acquisition mode
```

- The Byte (2) identify the Channel ID which are identified as below :

Channel	*Value*
Accelerometer Channel X	*0x01*
Accelerometer Channel Y	*0x02*
Accelerometer Channel Z	*0x00*
Inclinometer Channel X	*0x03*
Inclinometer Channel Y	*0x04*

Figure 11 :Channel ID

```
2: 0x1-----> channel X
```

- Bytes (3)(4)(5)(6) displays measurement time in UNIX format

```
3: 0xda
4: 0x4b
5: 0x4f
6: 0x5a

    > 5a4f4bda -----> GMT: Friday, January 5, 2018 9:56:42 AM
```

- Byte (7)(8)(9): Measurement Value (need to be divided by 1000)

```
7: 0xf
8: 0x0
9: 0x0

    > F (hex) ---> 15 (decimal)
    > Last bit is 0 so it's positive
    > 15/1000 = 0.015
```

2.4.2.2 Node-RED Advanced options

Now, after getting to receive our first data and understanding the content of a sample LowDutyCycle frame we proceed by making use of our Node-Red IoT tool and the countless possibilities it makes available.

2.4.2.2.1 Function

The function nodes will be the processing node of all the input data, in this example we used a function node to write our JavaScript line of codes to parse the payload frame as shown earlier, we have each info we need on its own.

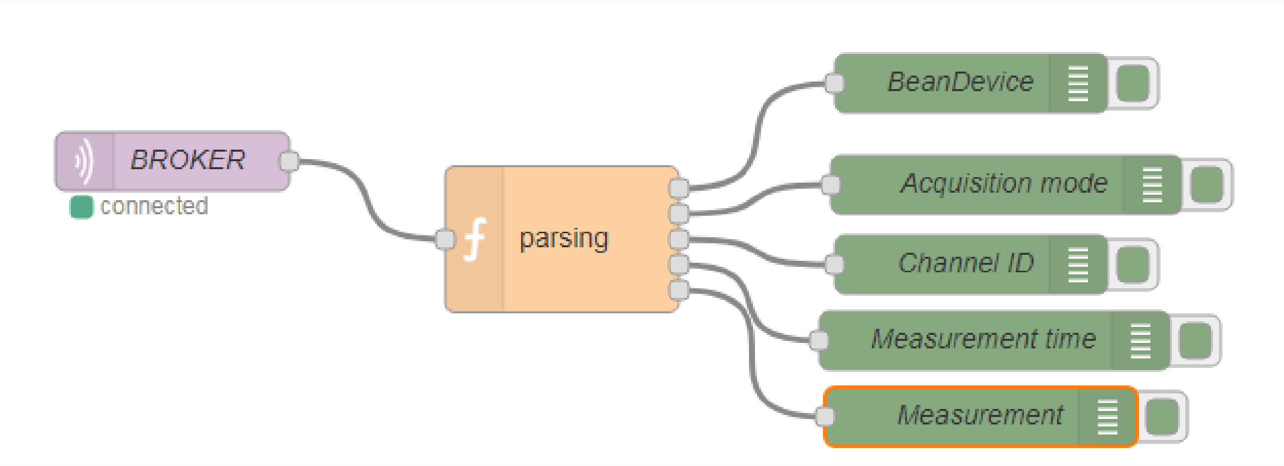


Figure 12 :Flow sample

We can then setup another function to listen for the measurement values and send notifications if a defined threshold is reached

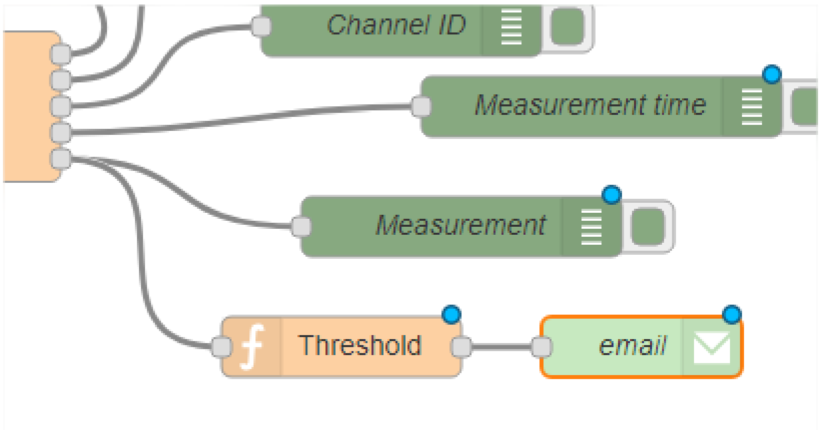


Figure 13 :Notification node

2.4.2.3 Notification nodes

Since notification is very important, you can use a SMTP node for notification,

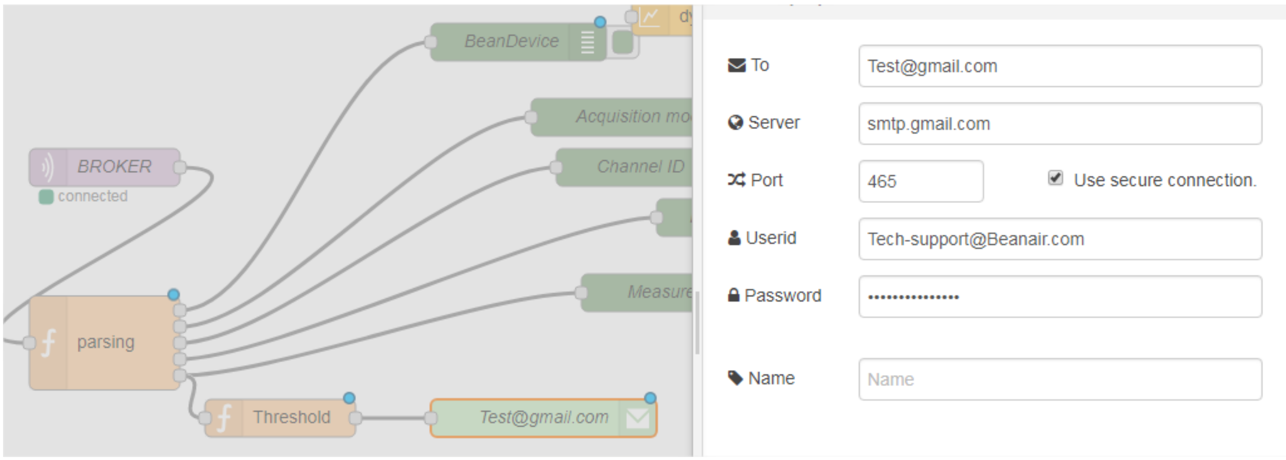


Figure 14 :SMTP setup

Or you can also use an online paid service to send SMS notifications to your mobile (using Twilio for example)

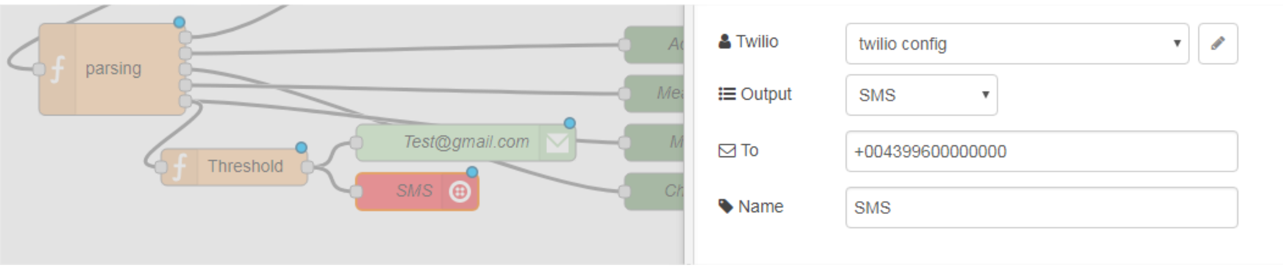


Figure 15 :SMS configuration

2.4.2.4 Display nodes

Displaying all of the signal and info is also possible using some default gauges, graphs, text holders, labels,etc. , or if available using some HTML/CSS skills will make a very good looking and useful dashboards to be accessed from a different Local URL

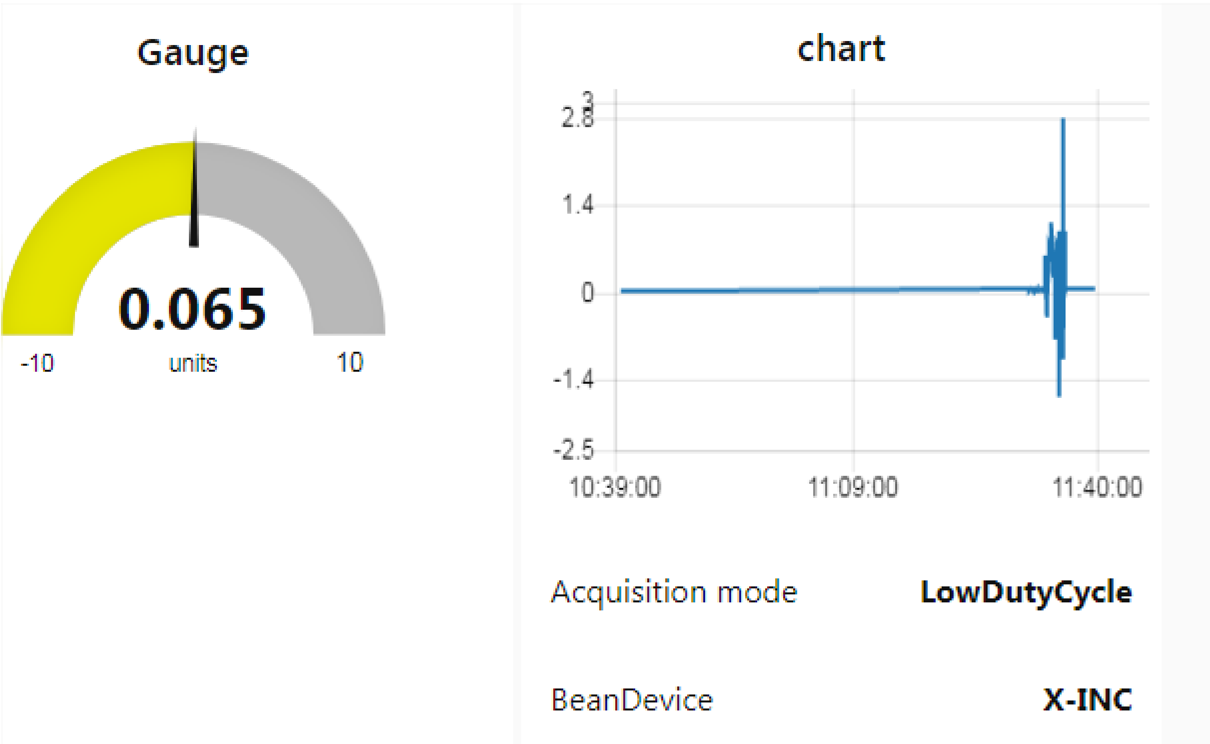


Figure 16 : Dashboard

2.4.2.5 Storage nodes

We can also use File storing nodes to store all the data locally

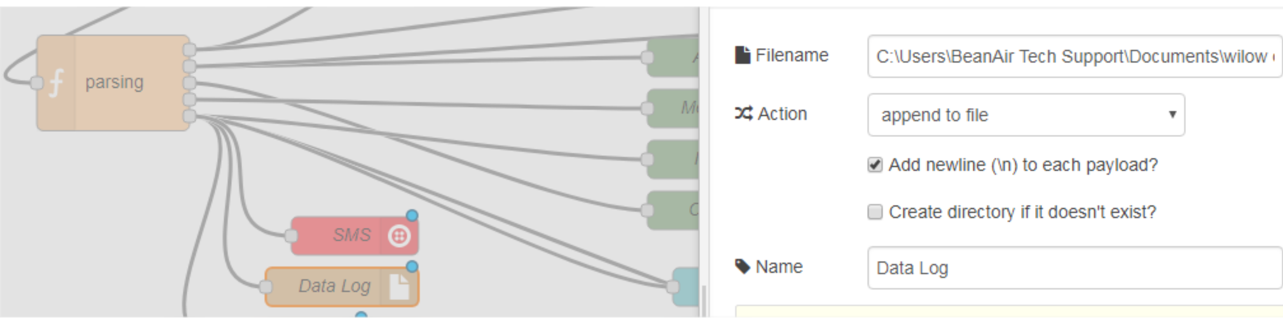


Figure 17 :Storage node

Or we can use local or online data base (MySQL for example) to manage all the data.

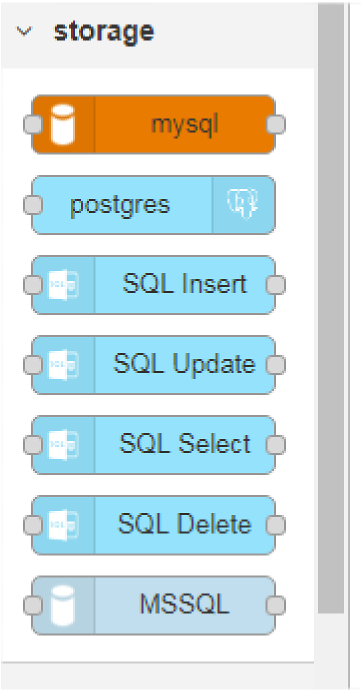


Figure 18 : Database nodes

2.4.3 Dynamic example

An example with NodeRed to collect data from Beandevice in streaming continue mode, which is one of the dynamic modes, will illustrate how to read, parse and extract all the information send from the device and how to manage to create a data visual application.

2.4.3.1 Subscribe to dynamic topic

For the dynamic mode we need to subscribe our Node-Red to the dynamic topic and it's only one topic for all channels and works for all dynamic modes (Streaming, SET, Shock Detection).

Topic name: [BeanDevice-MAC_ID]/STREAMING

Configure the MQTT node to listen to the streaming topic:

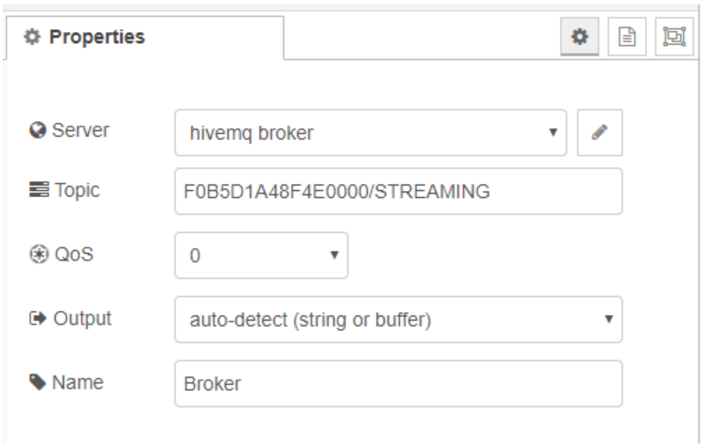


Figure 19: MQTT node configuration for dynamic mode

After you connect to broker and subscribe to the dynamic mode topic, you will be able to receive data from BeanDevice.(don't forget to start your device in streaming mode).

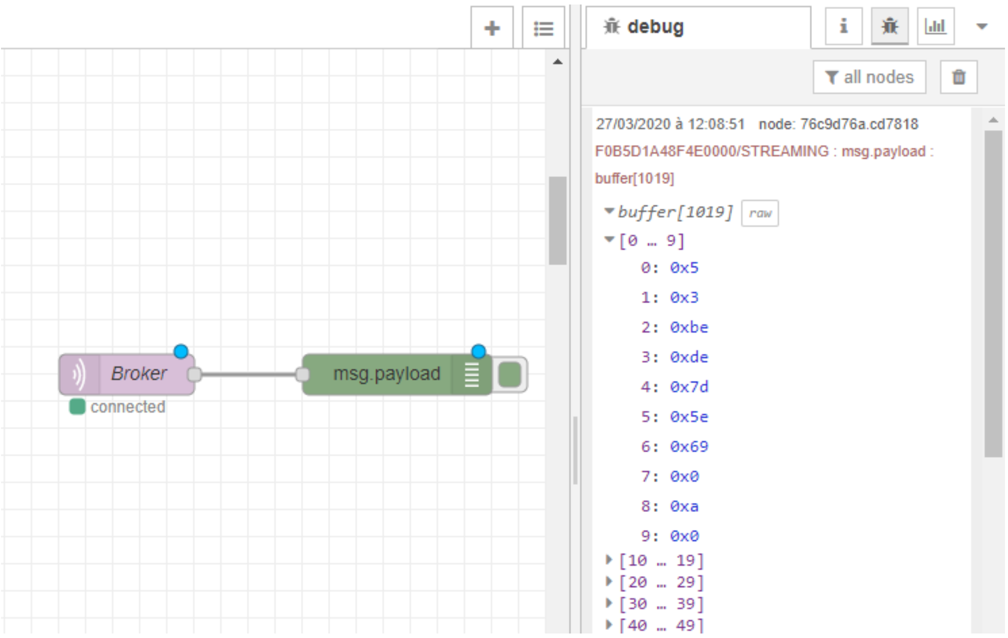


Figure 20: A simple debugger node displays the raw payload frame of Streaming mode

2.4.3.2 Payload

The MQTT frame sent from the BeanDevice® Wilow® in the Streaming mode will be composed in way that meets the high frequency publishing so the content will be distributed as follow:

Data meaning			Size	
Device Type			1 byte	
Acquisition type (Default 0x03)			1 byte	
Reference time In Unix time format (LSB First)			4 bytes	
Reference millisecond (LSB First)			2 bytes	
Sampling frequency (LSB First)			2 bytes	
Channels bitmap (LSB First)	Is channel 1 activated?	0th Bit	1st Byte	4 bytes
	Is channel 2 activated?	1st Bit		
	Is channel 3 activated?	2nd Bit		
	:	:		
	:	:		
	:	:	2nd Byte	
	:	:	3rd Byte	
	:	:	4th Byte	
	Is channel 32 activated ?	31th Bit		
Frame Sequence Id (LSB First):(Begins from 0)			3 bytes	
Number of data acquisitions per channel			2 bytes	
Data Acquisition cycle			3 bytes	
Data acquisition duration			3 bytes	
Previous Number of data acquisitions per channel			2 bytes	
Flags	Synchronization		1 bit	1 byte
	Future Use		7 bits	
Network Quality (LQI)			1 byte	

Data Sample 1 of channel 1 (LSB First)	Byte[0] data bits		1st Sub Packet	1 byte		3 bytes
	Byte[1]			1 byte		
	Byte[2]	Sign bit		8th bit	1 byte	
		data bits		7 bits		
:				:		
Data Sample 1 of channel n (last one present in the "channels bitmap" field) (LSB First)				3 bytes		
Data Sample 2 of channel 1 (LSB First)			2nd Sub Packet	3 bytes		
Data Sample 2 of next channel (LSB First)				3 bytes		
:				:		
Data Sample 2 of channel n (last one present in the "channels bitmap" field) (LSB First)				3 bytes		
:				:		
Data Sample M of channel 1 (LSB First)			Mth Sub Packet	3 bytes		
Data Sample M of next channel (LSB First)				3 bytes		
:				:		
Data Sample M of channel n (last one present in the "channels bitmap" field) (LSB First)				3 bytes		

Table 2: Streaming MQTT frame content

Example of a Streaming MQTT payload:

```
Device type: 0x05
Acquisition type: 0x03
Reference time: 0x07,0xE5,0x7D,0x5E
Reference millisecond: 0x15,0x00
Sampling frequency: 0x64,0x00
Channels bitmap: 0x1F,0x00,0x00,0x00
Frame sequence id: 0x12,0x00,0x00
Number of data per channel: 0x42,0x00
Data acquisition cycle: 0x00,0x00,0x00
Data acquisition duration: 0x00,0x00,0x00
Previous number of data per channel: 0x42,0x00
Flags: 0x01
Network quality: 0xE1
1st SubPacket:
Channel 1: 0xE2,0x03,0x00
Channel 2: 0x11,0x00,0x00
Channel 3: 0x14,0x00,0x80
Channel 4: 0x72,0x04,0x80
```

```
Channel 5: 0x6E,0x03,0x00
.
.
Mth SubPacket:
Channel 1: 0x10,0x00,0x80
Channel 2: 0x70,0x04,0x80
Channel 3: 0x6E,0x03,0x00
Channel 4: 0xDE,0x03,0x00
Channel 5: 0x0F,0x00,0x00
```

- Device type will be identified in the Byte (0) according to the table below:

Device type	Value	Description
AX_3D	0x01	AX_3D device Id
HI_INC_MONO	0x02	HI_INC_MONO Device Id
HI_INC_BI	0x03	HI_INC_BI Device Id
X_INC_MONO	0x04	X_INC_MONO Device Id
X_INC_BI	0x05	X_INC_BI Device Id
AX_3DS	0x06	AX_3DS device Id

Table 3: BeanDevice type

```
0: 0x5----> X-INC
```

- The Byte (1) will tell us about the Acquisition mode:

Data Acquisition type	Value	Description
LDCDA mode	0x01	The Id of the Low Duty Cycle Data Acquisition mode
Alarm mode	0x02	The Id of the Alarm Data Acquisition mode
Streaming mode	0x03	The Id of the Streaming Data Acquisition mode
Shock Detection mode	0x04	The Id of the Shock Detection mode
LDC Math Result	0x05	The Id of the Low Duty Cycle Math Result
SET mode	0x06	The Id of the SET (STREAMING WITH EVENT TRIGGER) mode
Dynamic Math Result	0x07	The Id of the Dynamic Math Result

Table 4: Acquisition modes

```
1: 0x1---->LowDutyCycle Acquisition mode
```

- Bytes (2)(3)(4)(5) displays measurement time in UNIX format (LSB first)

```
2: 0x07
3: 0xE5
4: 0x7D
5: 0x5E
```

> 5E7DE507 -----> 1585308935, GMT: Friday, March 27, 2020 12:35:35 AM

- Bytes (6)(7) displays millisecond part of the measurement time (LSB first)

```
6: 0x15
7: 0x00
```

> 0015 -----> 21 millisecond

- Bytes (8)(9) display sampling frequency (LSB first)

```
8: 0x64
9: 0x00
```

> 0064 -----> 100 hz

- Bytes (10)(11)(12)(13) display channels bitmap (LSB first), each bit represent the status of the channel. (if N^{th} bit 0 mean N^{th} channel is disabled and if it's 1 mean it's enabled)

```
10: 0x1F
11: 0x00
12: 0x00
13: 0x00
```

> 0000001F -----> channel 0, 1, 2, 3 , 4 and 5 are enabled

Channel	Value
Accelerometer Channel X	0x01
Accelerometer Channel Y	0x02
Accelerometer Channel Z	0x00
Inclinometer Channel X	0x03
Inclinometer Channel Y	0x04

Table 5: Channel ID

- Bytes (14)(15)(16) displays frame sequence id (LSB first)

```
14: 0x12
15: 0x00
16: 0x00
```

> 000012 -----> 18

- Bytes (17)(18) displays number of data per channel (LSB first)

```
17: 0x42
18: 0x00
```

> 0042 -----> 66 measurement per channel

- Bytes (19)(20)(21) displays data acquisition cycle (LSB first)

```
19: 0x00
20: 0x00
21: 0x00
```

> 000000 -----> 0: because we are using the streaming continue which require no acquisition cycle.

> Bytes (22)(23)(24) displays data acquisition duration (LSB first)

```
22: 0x00
23: 0x00
24: 0x00
```

> 000000 -----> 0: because the streaming continue doesn't require acquisition duration.

- Bytes (25)(26) displays the previous number of data per channel (LSB first)

```
25: 0x42
26: 0x00
```

> 0042 -----> 66

- Byte (27) displays flags: in the flags field only the first bit is used to represent the synchronization status and the rest are for future use.

```
27: 0x01
```

> 01 -----> bit 1 is 1: device synchronized

- Byte (28) displays network quality (LSB first)

```
28: 0xE1

    >  E1 -----> 225
```

- The rest of bytes represent the measurement values, each measurement in 3 bytes and the last bit is a sign bit (LSB first), and to get the correct value we need to device by 1000.

$$Decimal\ value = (-1)^{sign\ bit} * \frac{Remaining\ bits\ in\ decimal\ format}{1000}$$

Each SubPacket represent the acquired measurement for each channel at a specific time. To calculate that time we need to use this formula:

$$T_{SubPacket} = Reference\ Time\ Second + \frac{\text{Reference Millisecond}}{1000} + \left(\frac{1}{Sampling\ frequency}\right) * SubPacket\ Index$$

Where

$$SubPacket\ Index = (Frame\ Sequence\ Id * Previous\ Number\ of\ data\ acquisitions\ per\ channel) + Current\ SubPacket\ row$$

Because we have 5 channels, Each subPacket will hold 5 measurement.

```
1st SubPacket:
Channel 1:
29: 0xE2
30: 0x03
31: 0x00

    >  0003E2 -----> 994/1000 = 0.994 and last bit in 0 which mean
       it's a positive number.
    >  SubPacket index = 18 * 66 + 0 = 1188 (first SubPacket so
       the current SubPacket row is 0)
    >  TsubPacket = 1585308935 + 21/1000 + (1/100)*1188 =
       1585308946,901 -----> GMT: Friday 27 march 2020
       12:35:46.901
```

2.4.4 Node-RED Advanced options

After understanding the content of the MQTT streaming frame we can proceed to make use of the Node Red platform and make data more visible by being visual using graphs.

To make this happen we need to install some packages:

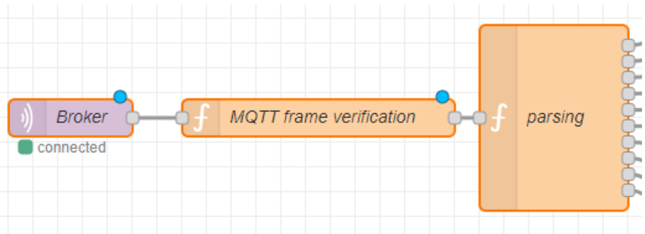
- Node-red-contrib-moment: to convert data/time from timestamp to human readable data/time. (link: https://github.com/TotallyInformation/node-red-contrib-moment)
- Node-red-dashboard: to create a live dashboard (graph) where we will display our measurements in real time. (link: https://github.com/node-red/node-red-dashboard)

2.4.4.1 Functions

- Verification and Parsing nodes:

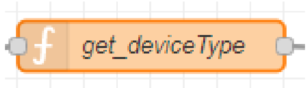
Before start parsing the MQTT frame we need to make sure it comes from a streaming mode, because the BeanDevice can publish MQTT frames for other acquisition modes like dynamic math result, SET and Shock detection, so we need to make sure to pass to our parsing node only rames from Streaming mode.

The parsing node function will receive as input the payload sent from the BeanDevice and captured by the Broker node and will slice it into sections, each section will hold only the payload of each field of the Streaming MQTT frame content and return them in array object.



- Device type node:

Based on the **Error! Reference source not found.** this node will return the type of the Beandevice i n readable format, String format.

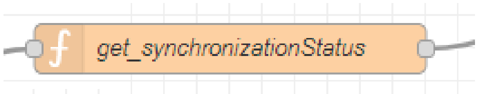


- Data acquisition mode node:

Based on the Table 4: Acquisition modes this node will return the acquisition mode in readable format, String format.

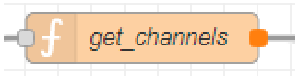
- Synchronization node:

Returns the synchronization status of the device.



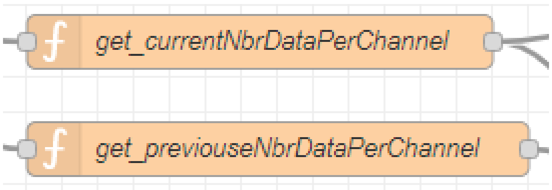
- List of channels node:

From the channels bitmap field, this function will determine which channel is enabled by using the bit index and with the help of Table 5: Channel ID we will return each active channel's name In Array format.



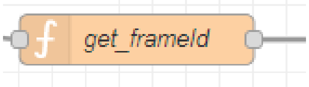
- Previous and Current number of data per channel node:

2 nodes with the same function, each one will calculate the number of measurement per channel using hex to decimal conversion, one for the previous number and the other for the current number.



- Frame Id node:

Return the frame Id after convert it from hex to decimal.



- Timestamp and Millisecond node:

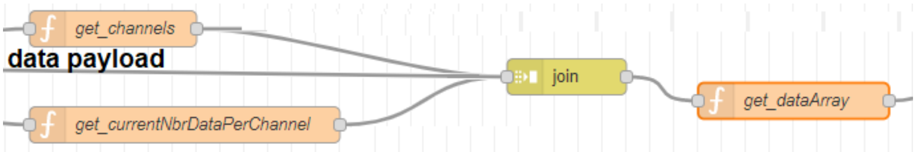
The time received from the BeanDeivce is divided in 2 fields, one for the timestamp (unix format) and the other for the milliseconds. This function will convert the timestamp from hex to decimal number and the same for the milliseconds and return both of them in Array format.

- Sampling frequency node:

Return the sampling frequency after convert it from hex to decimal.

- List of Data node:

To find out which measurement belong to which channel we need to know the number of data per channel and the number of active channels and by joining them with the data payload using join node we can pass those information to this function and with the help of Table 2: Streaming MQTT frame content we can return an array of 2 dimensions that store all converted data from hex to decimal.



- List of Time node:

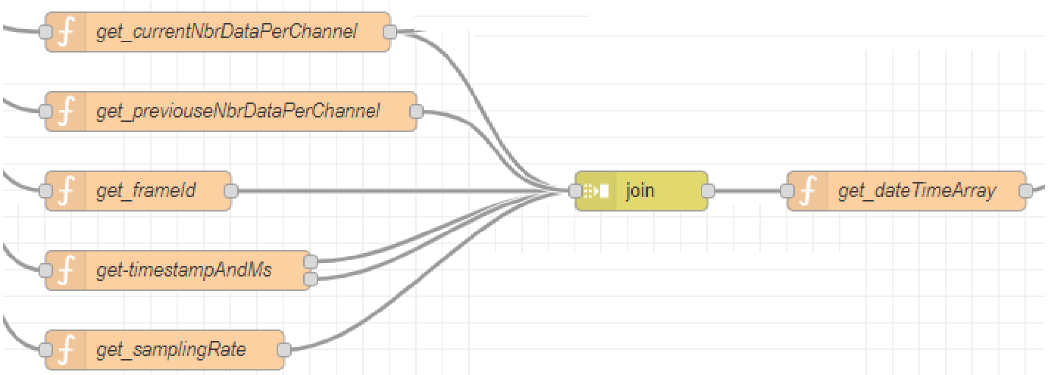
This function will calculate the corresponding time of each measurement by applying the formula:

$$T_{SubPacket} = Reference\ Time\ Second + \frac{\text{Reference Millisecond}}{1000} + \left(\frac{1}{Sampling\ frequency}\right) * SubPacket\ Index$$

Where

$$SubPacket\ Index = (Frame\ Sequence\ Id * Previous\ Number\ of\ data\ acquisitions\ per\ channel\ ) + Current\ SubPacket\ row$$

For this we need a join node to pass to our function the current and the previous number of data per channel, the frame id, the sampling rate and the timestamp with the millisecond part.



- To Dashboard nodes:

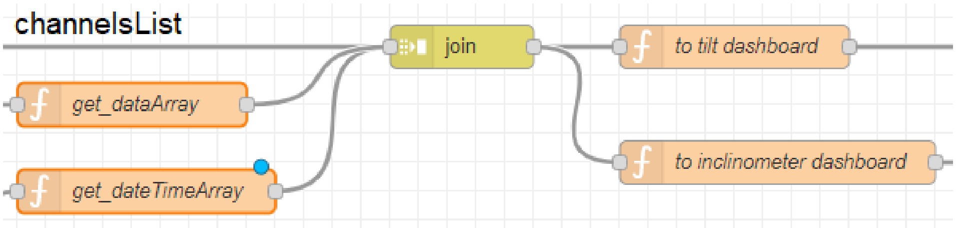
Before we send measurements to graph we need to prepare the data in a format that the dashboard can use it and because we are using "Node-red-dashboard" the data need to be in this format:

```
[{
"series": ["Channel 1", "Channel 2",.., "Channel n"],
"data": [
    [{"x": timestamp 1 in millisecond, "y": measurement 1 of Channel 1},
     {"x": timestamp 2 in millisecond, "y": measurement 2 of Channel 1},
        …
     {"x": timestamp m in millisecond, "y": measurement m of Channel 1}
    ],
    [{"x": timestamp 1 in millisecond, "y": measurement 1 of Channel 2},
     {"x": timestamp 2 in millisecond, "y": measurement 2 of Channel 2},
        …
     {"x": timestamp m in millisecond, "y": measurement m of Channel 2}
    ],
        …
    [{"x": timestamp 1 in millisecond, "y": measurement 1 of Channel n},
     {"x": timestamp 2 in millisecond, "y": measurement 2 of Channel n},
        …
     {"x": timestamp m in millisecond, "y": measurement m of Channel n}
    ]
],
"labels": [""]
}]

Our BeanDevice supports 2 sensor types (tilt and inclinometer), so it's better to use 2
graphs which require to create 2 nodes, both of them run the same code to prepare the
object we will feed to the dashboard's node but each one will use a different filter to
filter the upcoming data.
```

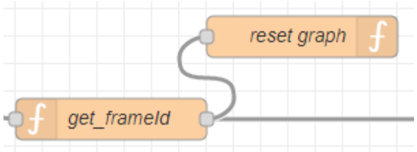
Note: because of the "Node-red-dashboard" timestamp to date/time conversion technique we need to multiply the timestamp by 1000 and use only the decimal part to display time in milliseconds.

To create a live chart we need to link the old data with the new one, so each time we receive a new frame we will store its data in a global variable within the flow context.



- Reset graph node:

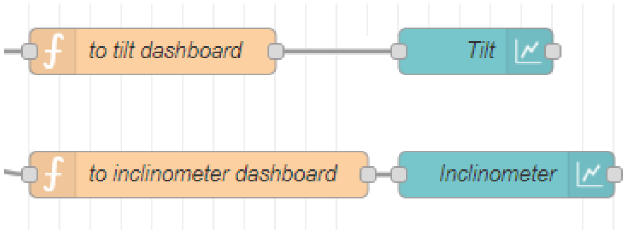
Streaming mode have 3 options (continue, burst and one shot), in burst option we need to refresh the graph each time a new streaming start, for this reason we need to empty the global variables each time we receive an MQTT frame with frame Id equal to 0.



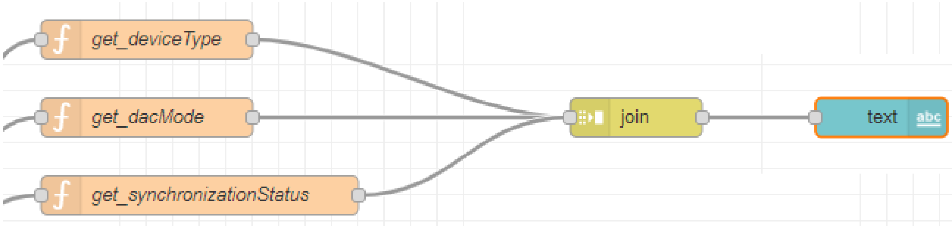
2.4.4.2 Dashboard

The Last step is to display our data in the graph, and to do this we need to use the chart node and the text node.

Chart node to display the graph for each sensor type.



Text node to display the device type and data acquisition mode with the synchronization status.



Note: to make sure we get the correct time with the time zone specified in Beandevice we need to change the host machine time zone to GMT.

2.4.4.3 Full Node Red Example

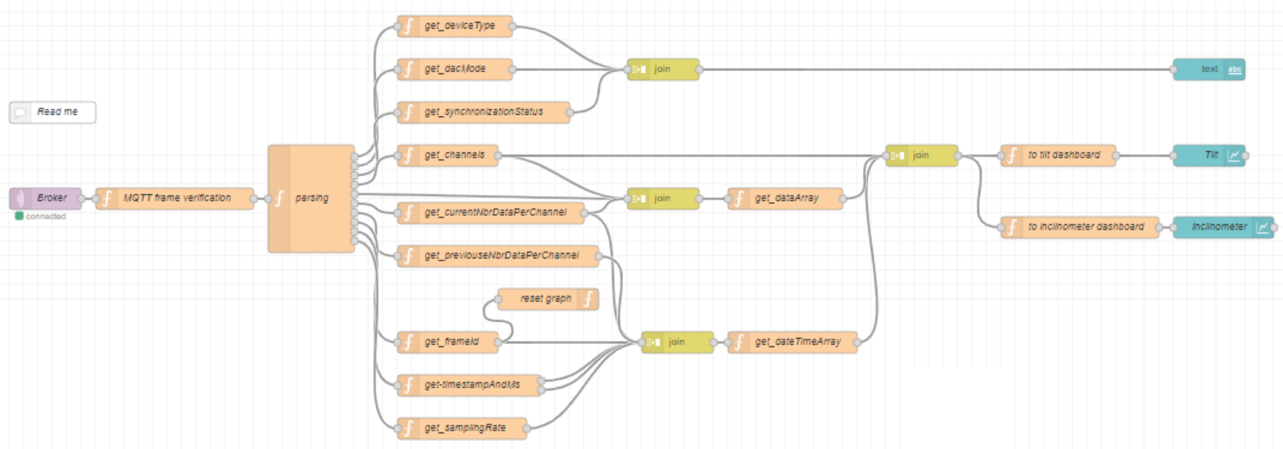


Figure 21: Full MQTT streaming client with Node Red

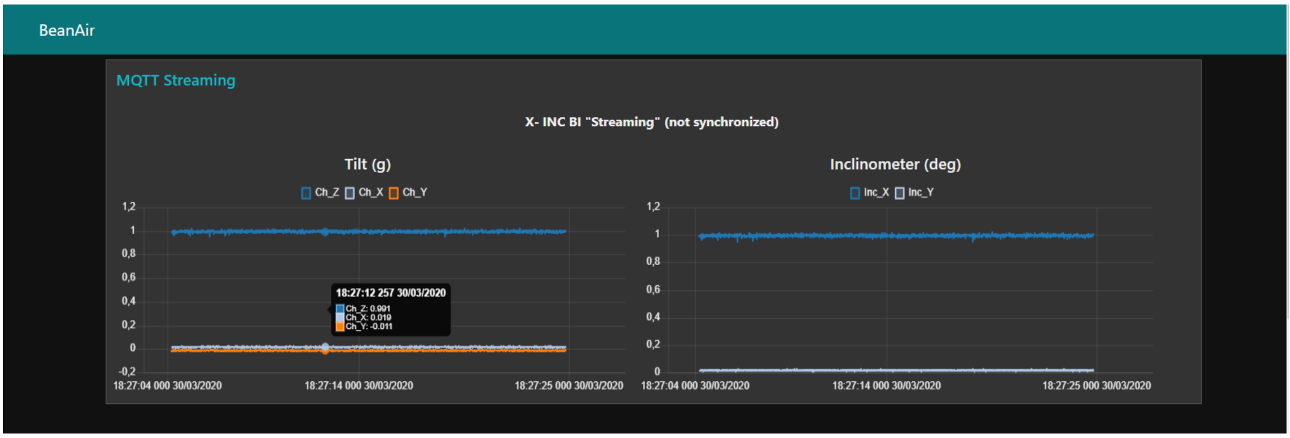


Figure 22: MQTT streaming client with node red results

2.5 CLOUD SOLUTION

The cloud solution requires a BeanDevice® Wilow® Connected to the internet, an available MQTT broker and any IoT cloud Platform, in this example we are going to use the IBM Cloud to setup and manage Our BeanDevice® Wilow®, this allows us to use a lot of API's to connect to other apps and services

https://console.bluemix.net/registration/?target=%2Fcatalog%2Fstarters%2Fnode-red-starter

You can use free of cost online MQTT brokers or paid one for better support, more connections and much bigger data rate, or You can Just use a locally installed MQTT broker.

After signing up go to your online Node-Red flow editor

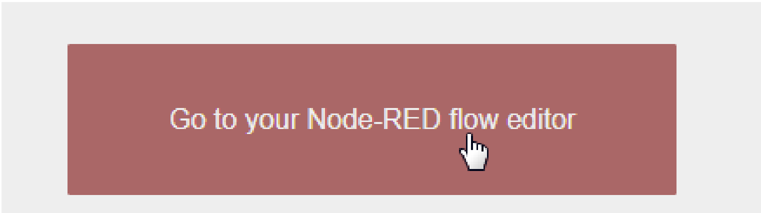


Figure 23 :Cloud connection

In the flow editor in this example and other than the same I did in the local setup example I managed to work on various flows to make sure all BeanDevice® Wilow® channels are monitored ,

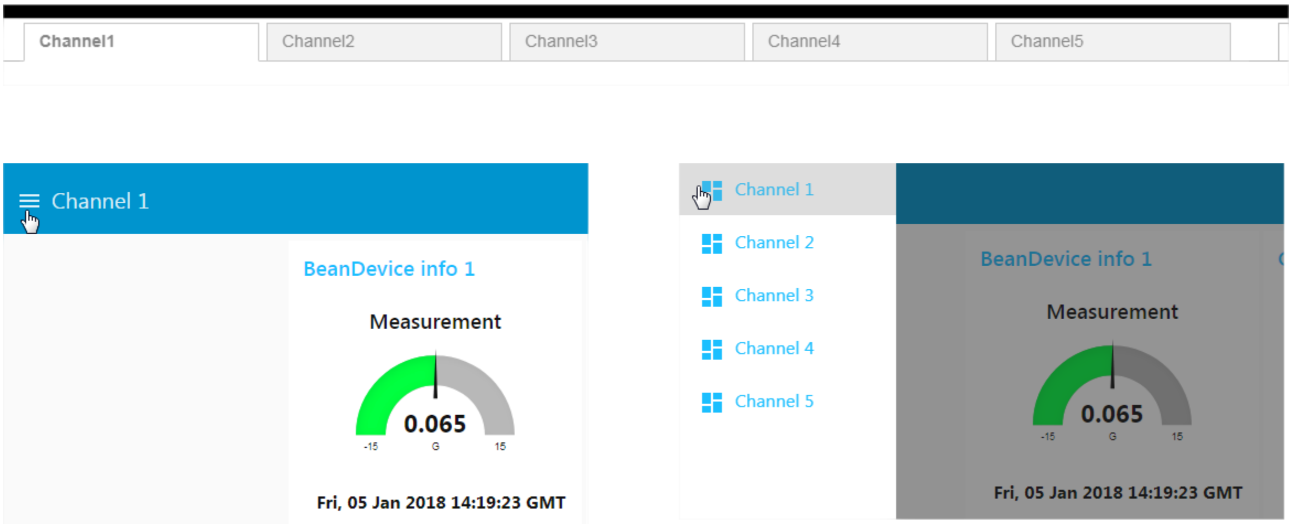


Figure 24 Display BeanDevice Channels on a web page

IBM Cloud allows you access to your monitoring address through a public customizable URL, for example:
https://wilow.eu-gb.mybluemix.net

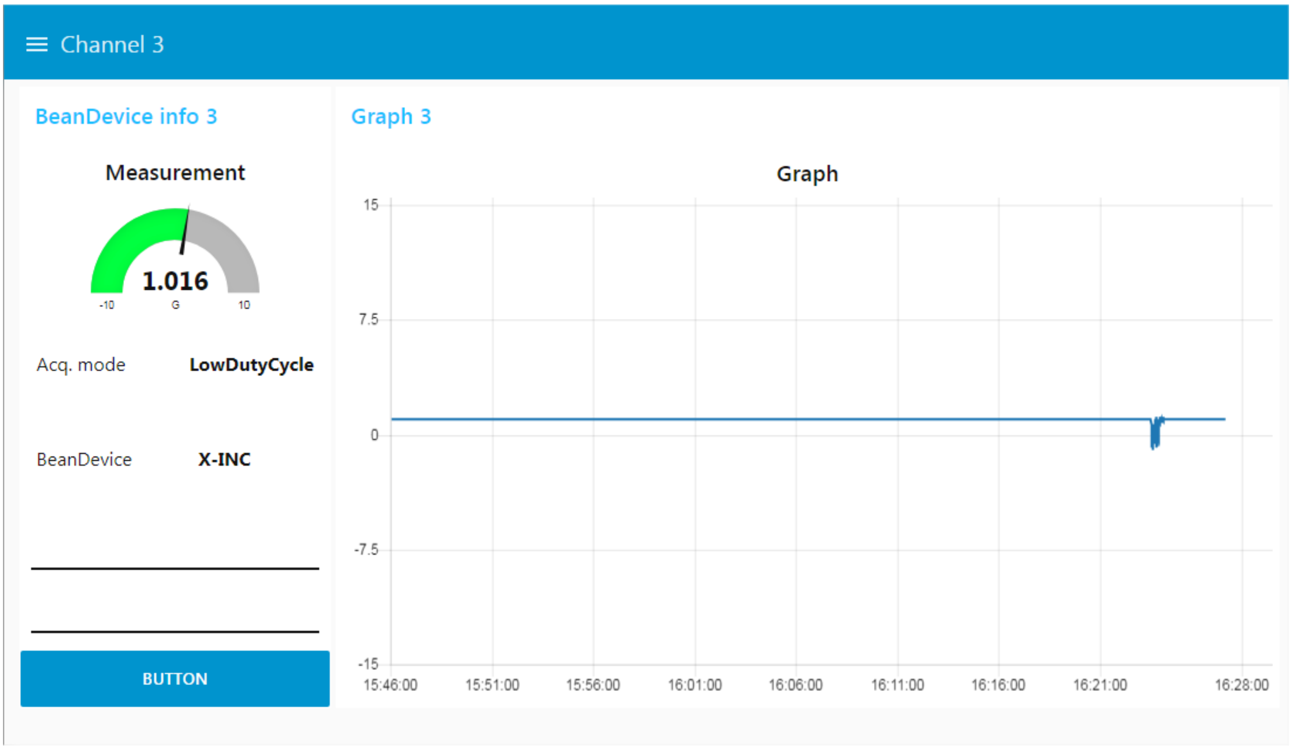


Figure 25 :Dashboard display on a web page

3. RELATED DOCUMENTS & VIDEOS

In addition to this technical note, please consult the related User guide, technical notes and videos:

Document name (Click on the web link)	Related product	Description
TN_RF_004 «MQTT Communication Protocol »	Wilow® products line	MQTT Communication Protocol for a seamless integration into a third-party IOT software
TN RF 005 «Building a reliable Wi-Fi network with Wilow sensors»	Wilow® products line	The aim of this document is to describe the autonomy performance of the BeanDevice® SmartSensor® and ProcessSensor® product line in streaming and streaming packet mode.
UM RF 007 «UM-RF-07-ENG-Wilow-Wifi-Sensor»	Wilow® products line	BeanDevice® Wilow® user manual

Beanair video link (YouTube)	Related products
Getting started with BeanDevice® Wilow - Wi-Fi Low Power Sensors	BeanDevice® Wilow
Wilow - Wi-Fi Sensors-Low duty cycle data acquisition mode on BeanDevice® Wilow	BeanDevice® Wilow
Wilow - Wi-Fi Sensors-Streaming mode with continuous monitoring on BeanDevice® Wilow	BeanDevice® Wilow
Wilow - Wi-Fi Sensors-How to setup Wilow Datalogger	BeanDevice® Wilow
Wilow - Wi-Fi Sensors-Smart Shock Detection (SSD) mode	BeanDevice® Wilow®

All the videos are available on our YouTube channel